HW2

黄佳溢

October 15, 2024

1 实验环境搭建

1.1 内核及文件系统加载

由于该部分和后面的内容有所交集，所以我们放在后面的章节展示。

1.2 交叉编译

由于实验室电脑是x86架构，而开发板使用龙芯架构，在开发时，我们需要使用龙芯架构的gcc编译器编译.c文件，生成的可执行文件才能在开发板上正常运行。

我们编写简易的hello world代码：

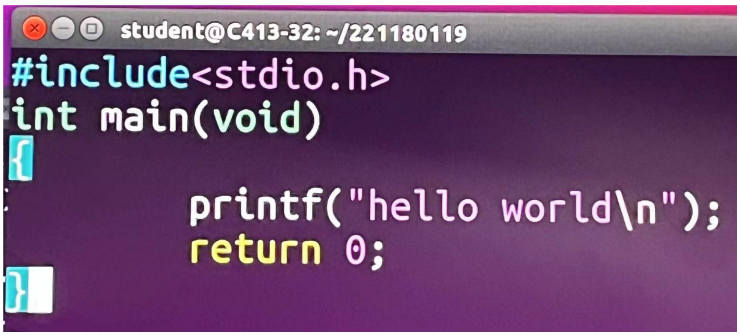

Figure 1: Hello world

下面在进行编译时，不能直接使用**gcc hello.c -o hello**，因为此时主机会使用x86架构的gcc编译，生成的可执行文件并不能被开发板所解析。为了解决这一点，我们需要使用交叉编译：

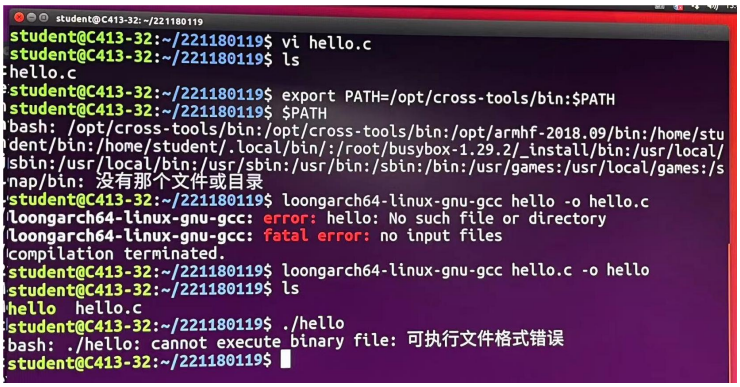

Figure 2: Gcc

可以看到，此时编译的hello可执行文件**不能**被主机执行，当我们将hello转移到被挂载的nfs4文件并在开发板上执行，**可以**执行！

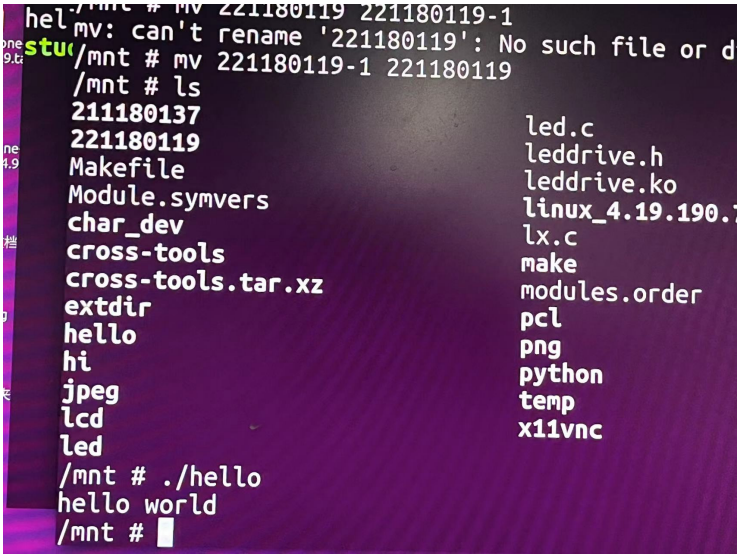

Figure 3: 开发板执行

2 内核和文件系统定制

2.1 默认内核制作

本次任务中，我需要熟悉内核和文件系统的制作。首先我们先来制作内核vmlinuz。以下是按照给定流程所呈现的结果：

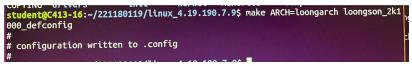

Figure 4: 步骤一

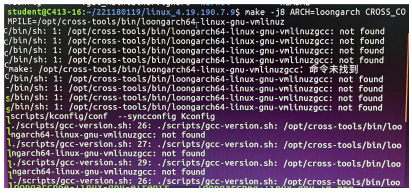

Figure 5: 步骤二

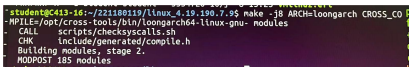

Figure 6: 步骤三

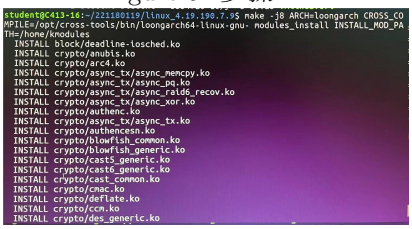

Figure 7: 步骤四

这里简单解释一下命令行：

1 由于linux_4.19.190.7.9文件可以制作多种内核，并不仅限于龙芯。所以在执行make命令时，需要注明ARCH=loongarch

2 我们执行第一步指令后，Makefile文件会将生成相应的配置文件.config

3 第二三四步指令是采用龙芯架构的gcc编译器进行编译时，需要指定编译器前缀，于是我们需要指定CROSS_COMPILE。执行make vmlinuz、modules和modules_install

完成这些之后，我们采用tftp协议，将生成的内核文件vmlinuz放置在/srv/tftpboot下，并在开发板中下载并加载，发现无法进入交互页面：

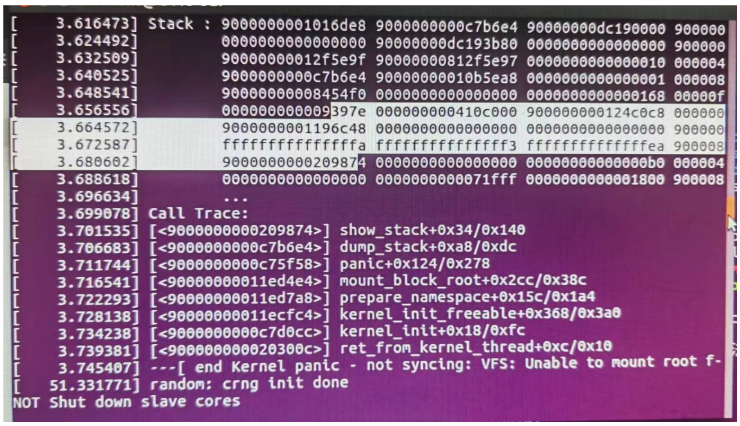

Figure 8: 加载vmlinuz

这是正常的，注意看倒数第二行的提示：**VFS: Unable to mount root f-**，这代表着我们缺少文件系统，无法建立unix系统的虚拟文件系统，也就无法进入交互页面。

2.2 定制内核制作

为了搭配定制文件系统，我们需要对内核的默认配置进行更改，由于本部分按照课件老老实实运行，基本没有出现问题，并且很多命令的解释都非常详实，这里我便只贴出部分步骤图片：

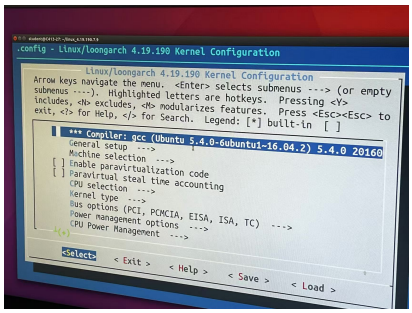

Figure 9: 步骤一

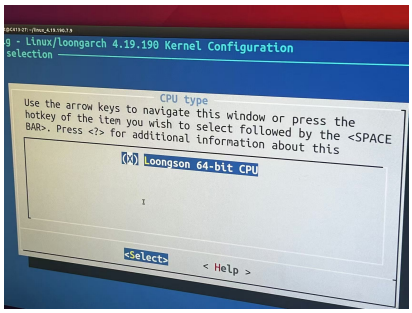

Figure 10: 步骤二

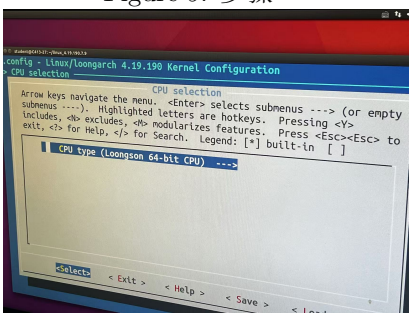

Figure 11: 步骤三

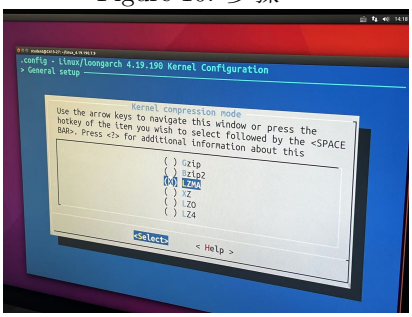

Figure 12: 步骤四

完成相关配置，执行ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu- make -j8 vmlinuz，编译内核文件，完成。

2.3 文件系统定制

这个部分是最卡住我们的部分，有很多踩坑的地方，会在后面的章节进行介绍。这里还是贴出我们的执行图片：

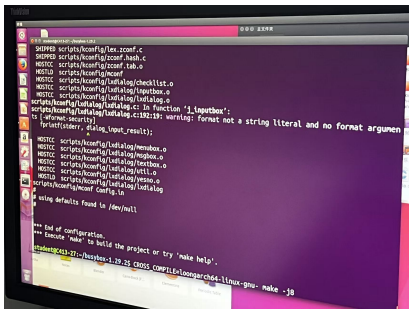

Figure 13: 步骤一

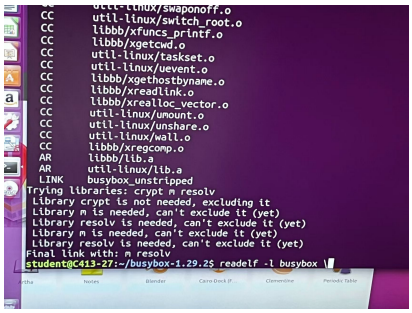

Figure 14: 步骤二

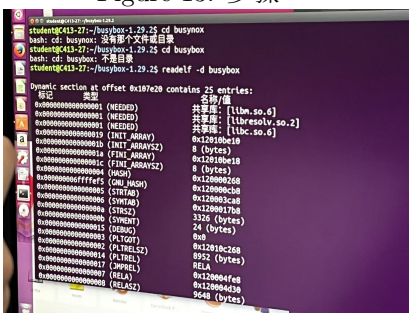

Figure 15: 步骤三

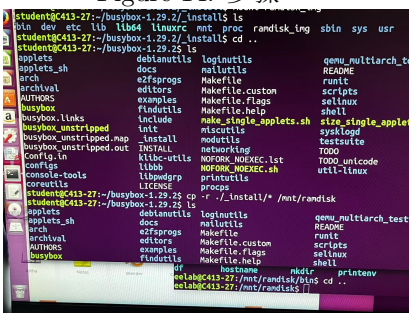

Figure 16: 步骤四

这里也就点一些注意事项:

1. 当我们需要链接共享库时，我们可以在某个可能的目录中执行**find -name ld-2.28.so**，该指令会告诉你哪里有。最好建议不要在根目录执行该指令，不然你会等它搜索半天。

2. 不要在_install目录下生成ramdisk_img，最好在home目录下，否则在后续的操作中，可能会出现**递归套用**的风险，因为你此时会将_install中的所有东西都拷贝进ramdisk_img里面，包括ramdisk_img本身。

3. 在操作中，我们需要将初始化好的ramdisk_img挂载到/mnt/ramdisk目录下，可能会出现报错，此时可以查看自己主机的/etc/fstab文件，看看有没有一行:

   ramdisk_img /mnt/ramdisk auto noauto ,users ,loop 0 0

   没有就去找老师开权限写。

差不多注意这些点，老老实实按照课件，效率会高一些。尽管如此，我们周围还是很多人会在最后开发板加载时失败（包括我们），原因也五花八门，这在下一章提及。

2.4 踩坑

1. **make ARcH=loongarch loongson 22k1000 defconfig报错**

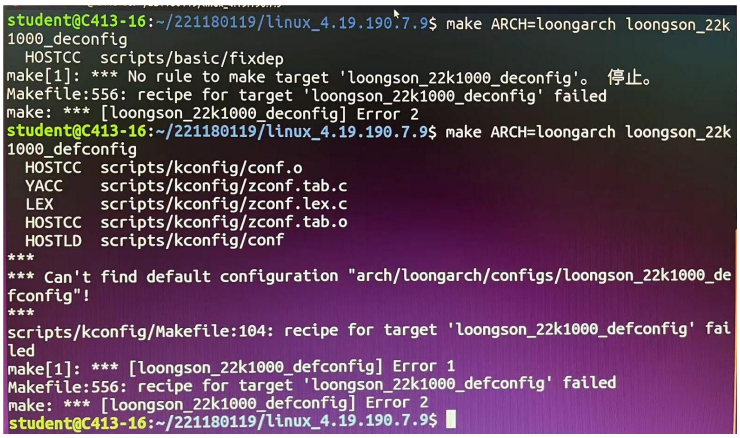

Figure 17: 踩坑一

解决方法：查看报错信息，发现是系统无法找到22k1000 defconfig文件，于是我们跟随报错路径进行查看：

发现此时并没有22k1000 defconfig文件，也就怪不得无法执行，此时将命令修改为make ARcH=loongarch loongson 2k1000 defconfig即可。

2. **第二条指令中，tools打成了toools**

3. **第三条指令中，gnu-和vmlinuz之间有空格，当时没注意，连在一起，导致无法寻找到该前缀！**

4. **开发板加载定制内核、文件系统报错**

这个是一个让我们很红温的事情，但幸运的是，我们的定制系统从头到尾只有一个bug，解决掉也就能正常运行了。下面分享一下，该bug的基本特征，以及如何识别并解决。

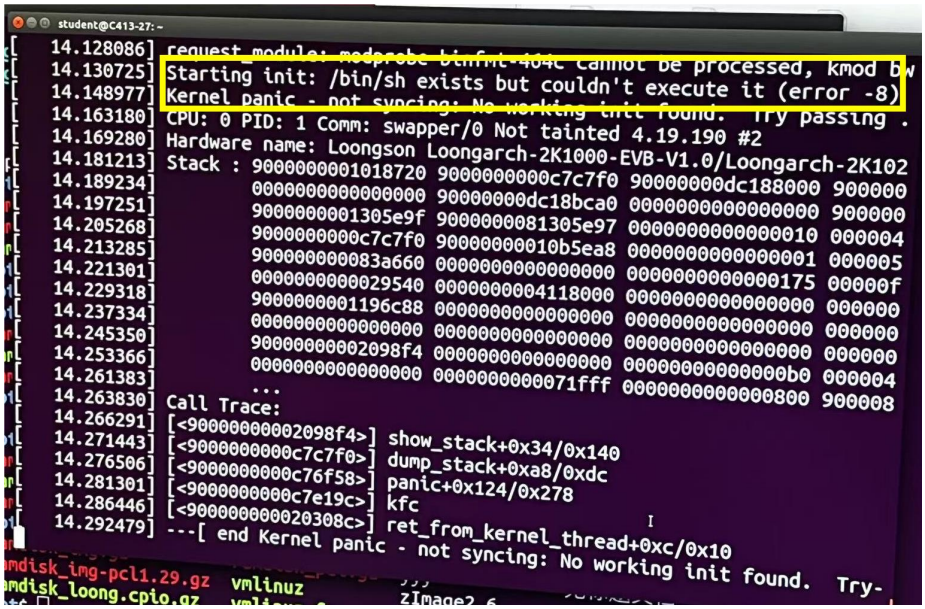

Figure 18: bug

很多时候我们不要被下面的**Kernel panic**吓唬到，需要往前找，注意到我们标黄色框框的地方，**/bin/sh exists but couldn't execute it**。

这种情况代表着你的文件系统可能不是**龙芯架构**。判断的方法便是：进入到_install/bin中，找到busybox文件，执行**file busybox**，就可以看出到底是什么架构的，一般来说，显示**unknown arch**应该就没啥问题。

假如和我们当时一样，显示**x86 arch**，那就是前面在定制文件系统时，编译或者下载的某一步没有加上**ARCH=loongarch**导致的，建议你删掉做好的ramdisk_img，重新按照教程，老老实实地做一次。

3 总结致谢

最后，我得说，大多数同学都会卡在最后一步无法进入交互界面，我还看到别的同学那种，没有ram0等等各种各样的bug，他们最终是否解决，如何解决都不得而知，遇到问题还是多去找老师，少红温，放平心态才是最重要的！

最后的最后，感谢我的好舍友吕展航和尤智宣同学对我的帮助和支持！感谢实验室的老师和助教们！